8-2019

Adversarial learning on heterogeneous information networks

Binbin HU

Yuan FANG
Singapore Management University, yfang@smu.edu.sg

Chuan SHI

DOI: https://doi.org/10.1145/3292500.3330970

Adversarial Learning on Heterogeneous Information Networks

Binbin Hu[1,2†], Yuan Fang[3], Chuan Shi[1‡]
[1]Beijing University of Posts and Telecommunications
[2] AI Department, Ant Financial Services Group
[3]Singapore Management University
{hubinbin,shichuan}@bupt.edu.cn,yfang@smu.edu.sg

ABSTRACT

Network embedding, which aims to represent network data in a low-dimensional space, has been commonly adopted for analyzing heterogeneous information networks (HIN). Although exiting HIN embedding methods have achieved performance improvement to some extent, they still face a few major weaknesses. Most importantly, they usually adopt negative sampling to randomly select nodes from the network, and they do not learn the underlying distribution for more robust embedding. Inspired by generative adversarial networks (GAN), we develop a novel framework HeGAN for HIN embedding, which trains both a discriminator and a generator in a minimax game. Compared to existing HIN embedding methods, our generator would learn the node distribution to generate better negative samples. Compared to GANs on homogeneous networks, our discriminator and generator are designed to be *relation-aware* in order to capture the rich semantics on HINs. Furthermore, towards more effective and efficient sampling, we propose a *generalized* generator, which samples "latent" nodes directly from a continuous distribution, not confined to the nodes in the original network as existing methods are. Finally, we conduct extensive experiments on four real-world datasets. Results show that we consistently and significantly outperform state-of-the-art baselines across all datasets and tasks.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Learning latent representations**.

KEYWORDS

Heterogeneous Information Network; Network Embedding; Generative Adversarial Network

ACM Reference Format:
Binbin Hu, Yuan Fang, Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA. ACM, NY, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330970

† Work done while a visiting research student at Singapore Management University.
‡ The corresponding author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330970

1 INTRODUCTION

Network structures are ubiquitous in real-world applications, ranging from social and biological networks to transportation and telecommunication systems. Thus, network analysis is becoming increasingly important towards solving crucial problems such as personalized user recommendation on social networks [41] and disease gene identification on biological networks [2]. These problems often unfold as instances of node clustering, node classification and link prediction on network data, which fundamentally depend on an effective form of network representation. In recent years, network embedding [3, 8] has emerged as a promising direction for unsupervised learning of node representations, which aims to project the nodes of a network into a low-dimensional space whilst preserving the structural properties of the original network.

Heterogeneous information networks. While earlier network embedding work [14, 23] has attained considerable success, they can only cope with the so-called homogeneous networks, which comprise one single type of nodes and edges. However, in real-world scenarios, nodes naturally model different types of entity, which interact with each other through multiple relations. Such networks are known as *Heterogeneous Information Networks* (HIN) [28], as illustrated in Fig. 1(a) for bibliographic data. Observe that the toy HIN consists of multiple types of node (*e.g.,* author and paper), which are connected by various types of relation (*e.g.,* write/written relation between author and paper, and publish/published relation between paper and conference).

Due to its heterogeneity, HINs often carry immensely rich and complex semantics. Thus, more recent research has shifted towards HIN embedding, most notably Metapath2vec [11] and HIN2vec [12]. As shown in Fig. 1(b-1), existing HIN embedding methods fundamentally boil down to two samplers, which select "context" nodes from the network as positive (*e.g.,* author a_2) and negative (the shaded circles) examples for a given "center" node (*e.g.,* paper p_2), respectively. (Note that each node can serve as the center or context similar to the Skip-gram model [21].) Subsequently, a loss function is trained on these samples to optimize node representations. Although these methods have achieved some performance improvement, they suffer from serious limitations. First, they commonly utilize negative sampling to randomly select existing nodes in the network as negative samples. As such, their negative samples are not only arbitrary but also confined to the universe of the original

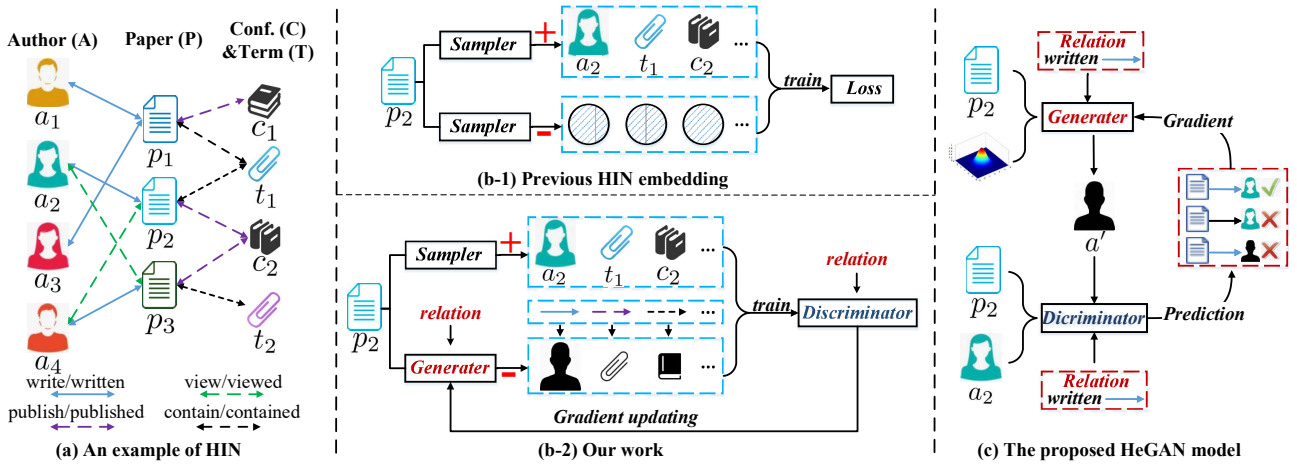


Figure 1: Overview of our work. (a) A toy example of HIN for bibliographic data. (b) Comparison between our work and previous works. (c) The framework of our proposed model HeGAN for adversarial learning on HINs.

network. Second, they mainly focus on capturing the rich semantics on HINs without heeding to the underlying distribution of the nodes, and thus lack robustness for real-world HINs that are often sparse and noisy. Third, many HIN embedding methods [11, 26] rely on appropriate meta-paths to match the desired semantics, which typically require domain knowledge that is sometimes subjective and often expensive to obtain.

Adversarial learning. On another line, *Generative Adversarial Networks* (GAN) [13, 25] have been developed for learning robust latent representations in various applications [10, 35, 37]. GANs hinge on the idea of adversarial learning, where a discriminator and generator compete with each other to not only train a better discriminative model, but also learn the underlying data distribution. The latter makes the model more robust to sparse or noisy data [13, 24], and also provides better samples to reduce the labeling requirement. Given these advantages, there have been a few initial efforts on GAN-based network embedding [9, 22, 33, 38]. However, these studies only investigate homogeneous networks and thus do not account for the heterogeneity of nodes and edges, resulting in unsatisfactory performance on semantic-rich HINs.

Present work and challenges. Given the above limitations in current approaches, in this paper, we exploit the heterogeneity of HINs in an adversarial setting in order to learn semantic-preserving and robust node representations. However, its materialization is non-trivial, given two major challenges not addressed in existing GAN-based methods for homogeneous networks.

First, how to capture the semantics of multiple types of nodes and relations? In existing methods, real (*i.e.,* positive) and fake (*i.e.,* negative) nodes are only differentiated by network structure. Thus, it is imperative to design new forms of discriminator and generator that can differentiate and model real and fake semantic-rich nodes involved in various relations.

Second, how to generate fake samples efficiently and effectively? In existing methods, generators learn a finite discrete distribution over the nodes in a network. Thus, they often need to compute the intractable softmax function, and ultimately resort to approximations such as negative sampling [9] or graph softmax [33]. Moreover, they are essentially picking an existing node from the original network according to the learned distribution, without the ability to generalize to "unseen" nodes. Not surprisingly, they do not generate the most representative fake nodes, as such nodes may not even appear in the network. Thus, it is important to design a generator that can efficiently produce latent fake samples.

Insights and contributions. To address the above challenges, we propose **HeGAN**, a novel framework for HIN embedding with GAN-based adversarial learning. In particular, we propose a new form of discriminator and generator, as illustrated in Fig. 1(b-2). For the first challenge, our discriminator and generator are designed to be *relation-aware* in order to distinguish nodes connected by different relations. That is, *w.r.t.* any relation, the discriminator can tell whether a node pair is real or fake, whereas the generator can produce fake node pairs that mimic real pairs. In particular, a node pair is considered real only if (i) it is a positive pair based on network topology; and (ii) the pair is formed under the correct relation. For the second challenge, we design a *generalized* generator, which is able to directly sample latent nodes from a continuous distribution, such that (i) no softmax computation is necessary; and (ii) fake samples are not restricted to the existing nodes.

In summary, we make the following contributions.

(1) We are the first to employ adversarial learning for HIN embedding, in order to utilize the rich semantics on HINs. The solution is non-trivial due to the heterogeneity and the need for efficient and effective sample generation.

(2) We propose a novel framework HeGAN that is not only relation-aware to capture rich semantics, but also equipped with a generalized generator that is effective and efficient.

(3) We perform experiments on four public datasets on a series of downstream tasks. Results demonstrate that HeGAN consistently and significantly outperform various state-of-the-arts.

Table 1: Summary of notations

Notion	Explanation
$\mathcal{G}$	a heterogeneous information network (HIN)
$\mathcal{V}, \mathcal{E}$	the set of node and edges, resp.
$\mathcal{A}, \mathcal{R}$	the set of node and edge types, resp.
$\mathbf{e}^G, \mathbf{e}^D$	node embedding of generator and discriminator, resp.
$\mathbf{M}^G, \mathbf{M}^D$	relation matrix of generator and discriminator, resp.
θ^G, θ^D	parameters of generator and discriminator, resp.

2 PRELIMINARY

In this section, we formalize the problem of HIN embedding and introduce the background on GAN. Main notations are summarized in Table 1, including those to be introduced in Sect. 3.

HIN embedding. Our study focuses on heterogeneous information networks [28], which can be defined as follows.

Definition 1. ***Heterogeneous information network (HIN).*** *A HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi\}$ is a form of graph, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. It is also associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the sets of node and edge types such that $|\mathcal{A}| + |\mathcal{R}| > 2$.*

An example of HIN is illustrated in Fig. 1(a) on bibliographic data. Observe that it consists of multiple node types (*e.g.,* author, paper, conference and term) and their semantic relations (*e.g.,* author–write–paper, paper–contain–term).

The goal of HIN embedding is to learn a mapping function to project each node $v \in \mathcal{V}$ to a low-dimensional space $\mathbb{R}^d$, where $d \ll |\mathcal{V}|$. Node representations in the new space should preserve not only the structure, but also the rich semantics on the HIN.

Generative adversarial networks. Our work is inspired by the recent success of GANs, which can be viewed as a minimax game between two players, namely, generator G and discriminator D, in the following manner.

$$\min_{\theta^G} \max_{\theta^D} \quad \mathbb{E}_{x \sim P_{\text{data}}} \left[\log D(x; \theta^D)\right] + \mathbb{E}_{z \sim P_Z} \left[\log\left(1 - D(G(z; \theta^G); \theta^D)\right)\right]. \quad (1)$$

The generator G tries to generate fake samples as close to the true data as possible with the noise z from a predefined distribution P_Z, where θ^G denotes the parameters of the generator. On the contrary, the discriminator D aims to distinguish real data from the distribution P_{data} and fake data from the generator, where θ^D denotes the parameters of the discriminator. In practice, GAN has been found [25] to often work better if the generator minimizes $-\log D(G(\cdot; \theta^G); \theta^D)$ instead of $\log\left(1 - D(G(\cdot; \theta^G); \theta^D)\right)$.

3 THE PROPOSED APPROACH: HeGAN

In this section, we present the proposed model HeGAN, a novel HIN embedding approach based on GAN. We begin with the overall framework, followed by elaborations on our discriminator and generator. Lastly, we discuss the optimization of our framework, as well as comparison to other models.

3.1 Overall Framework of HeGAN

As shown in Fig. 1(c), our framework mainly consists of two competing players, the discriminator and the generator. Given a node, the generator attempts to produce fake samples associated with the given node to feed into the discriminator, whereas the discriminator tries to improve its parameterization to separate the fake samples with the real ones actually connected to the given node. The better trained discriminator would then force the generator to produce better fake samples, and the process is repeated. During such iterations, both the generator and discriminator receives mutual, positive reinforcement. While this setup may appear similar to previous efforts [4, 9, 22, 33, 38] on GAN-based network embedding, we employ two major novelties to address the challenges of adversarial learning on HINs.

First, existing studies only leverage GAN to distinguish whether a node is real or fake *w.r.t.* structural connections to a given node, without accounting for the heterogeneity in HINs. For example, given a paper p_2, they treat nodes a_2, a_4 as real, whereas a_1, a_3 are fake simply based on the topology of the HIN shown in Fig. 1(a). However, a_2 and a_4 are connected to p_2 for different reasons: a_2 writes p_2 and a_4 only views p_2. Thus, they miss out on the valuable semantics carried by HINs, unable to differentiate a_2 and a_4 even though they play distinct semantic roles. Towards semantic-preserving embedding, we introduce a *relation-aware* discriminator and generator, to differentiate various types of semantic relation between nodes. On our toy HIN, given a paper p_2 as well as a relation, say, write/written, our discriminator is able to tell apart a_2 and a_4, and our generator will try to produce fake samples more similar to a_2 instead of a_4.

Second, existing studies are limited in sample generation in both effectiveness and efficiency. They typically model the distribution of nodes using some form of softmax over all nodes in the original network. In terms of effectiveness, their fake samples are constrained to the nodes in the network, whereas the most representative fake samples may fall "in between" the existing nodes in the embedding space. For example, given a paper p_2, they can only choose fake samples from $\mathcal{V}$, such as a_1 and a_3. However, both may not be adequately similar to real samples such as a_2. Towards a better sample generation, we introduce a *generalized* generator that can produce latent nodes such as a' shown in Fig. 1(c), where it is possible that $a' \notin \mathcal{V}$. For instance, a' could be the "average" of a_1 and a_3 and is more similar to the real sample a_2. In terms of efficiency, the softmax function is expensive to compute, and approximations such as negative sampling and Graph Softmax must be employed. In contrast, our generator can sample fake nodes directly without using a softmax.

3.2 Discriminator and Generator in HeGAN

In the ensuing discussion, we shall zoom into the proposed discriminator and generator for HeGAN .

3.2.1 Relation-aware discriminator. As motivated, on a HIN it is imperative to distinguish real and fake nodes under a given relation. Thus, our relation-aware discriminator $D(e_v|u, r; \theta^D)$ evaluates the connectivity between the pair of nodes u and v *w.r.t.* a relation r. Specifically, $u \in \mathcal{V}$ is a given node and $r \in \mathcal{R}$ is a given relation from the HIN $\mathcal{G}$, e_v is the embedding of a sample node v (which

can be real or fake), and θ^D denotes the model parameters of D. In essence, D outputs a probability that the sample v is connected to u under the relation r. We quantify this probability as:

$$D(\mathbf{e}_v|u,r;\theta^D) = \frac{1}{1+\exp\left(-\mathbf{e}_u^{D\top}\mathbf{M}_r^D\mathbf{e}_v\right)}, \tag{2}$$

where $\mathbf{e}_v \in \mathbb{R}^{d\times 1}$ is the input embedding of the sample v, $\mathbf{e}_u^D \in \mathbb{R}^{d\times 1}$ is the learnable embedding of node u, and $\mathbf{M}_r^D \in \mathbb{R}^{d\times d}$ is a learnable relation matrix for relation r. $\theta^D = \{\mathbf{e}_u^D : u \in \mathcal{V}, \mathbf{M}_r^D : r \in \mathcal{R}\}$ form the model parameters of D, *i.e.,* the union of all node embeddings and relation matrices learnt by D.

Naturally, the probability should be high when v is a positive sample related to u through r, or low when it is a negative sample. In general, a sample v form a triple $\langle u, v, r\rangle$ together with the given u and r, and each triple belongs to one of the three cases below with regard to its polarity. Each case also contributes to one part of the discriminator's loss, inspired by the idea of conditional GAN [25].

Case 1: Connected under given relation. That is, nodes u and v are indeed connected through the right relation r on the HIN $\mathcal{G}$, such as $\langle a_2, p_2, \text{write}\rangle$ shown in Fig. 1(a). Such a triple is considered positive and can be modeled by the below loss.

$$\mathcal{L}_1^D = \mathbb{E}_{\langle u,v,r\rangle\sim P_{\mathcal{G}}} - \log D(\mathbf{e}_v^D|u,r). \tag{3}$$

Here we draw the positive triple from the observed $\mathcal{G}$, denoted as $\langle u, v, r\rangle \sim P_{\mathcal{G}}$.

Case 2: Connected under incorrect relation. That is, u and v are connected in the HIN under a wrong relation $r' \neq r$, such as $\langle a_2, p_2, \text{view}\rangle$. The discriminator is expected to mark them as negative too, as their connectivity does not match the desired semantics carried by the given relation r. We define this part of loss as follows.

$$\mathcal{L}_2^D = \mathbb{E}_{\langle u,v\rangle\sim P_{\mathcal{G}}, r'\sim P_{\mathcal{R}'}} - \log\left(1 - D(\mathbf{e}_v^D|u,r')\right). \tag{4}$$

Here, we still draw the pair of nodes $\langle u, v\rangle$ from $\mathcal{G}$, but the negative relation r' is drawn from $\mathcal{R}' = \mathcal{R} \setminus \{r\}$ uniformly.

Case 3: Fake node from relation-aware generator. That is, given a node $u \in \mathcal{V}$, it can form a fake pair with the node v supplied by the generator $G(u, r; \theta^G)$, such as $\langle a', p_2, \text{write}\rangle$ in Fig. 1(c). As we shall see later in Sect. 3.2.2, the generator is also relation-aware: it attempts to generate a fake node's embedding that mimics the real nodes connected to u under the correct relation r. Again, the discriminator aims to identify this triple as negative, which can be formulated as follows.

$$\mathcal{L}_3^D = \mathbb{E}_{\langle u,r\rangle\sim P_{\mathcal{G}}, \mathbf{e}'_v\sim G(u,r;\theta^G)} - \log\left(1 - D(\mathbf{e}'_v|u,r)\right). \tag{5}$$

Note that the fake sample v's embedding $\mathbf{e}'_v$ is drawn from the generator G's learnt distribution. As we shall see in Sect. 3.2.2, it is distinct from G's own model parameters θ^G. On the other hand, the discriminator D simply treats $\mathbf{e}'_v$ as non-learnable input, and only optimizes its own parameters θ^D.

It is worth noting that, arguably there could be a fourth case, where the triple is a fake pair from a relation-oblivious generator. However, such negative triples are believed to be more easily separated from the positives than the negatives in Case 2 or 3. Thus, we do not consider them here, although our model is flexible to incorporate this fourth case.

We integrate the above three parts to train the discriminator:

$$\mathcal{L}^D = \mathcal{L}_1^D + \mathcal{L}_2^D + \mathcal{L}_3^D + \lambda^D\|\theta^D\|_2^2, \tag{6}$$

where $\lambda_D > 0$ controls the regularization term to avoid overfitting. The parameters θ^D of the discriminator can be optimized by minimizing $\mathcal{L}^D$.

3.2.2 Relation-aware, generalized generator. The goal of our generator $G(\cdot\,;\theta^G)$ is to generate fake samples to mimic the real one. On the one hand, G is relation-aware just as the discriminator. Thus, given a node $u \in \mathcal{V}$ and a relation $r \in \mathcal{R}$, the generator $G(u, r; \theta^G)$ aims to generate a fake node v likely to connect to u in the context of relation r. In other words, v should be as close as possible to a real node, say, w such that $\langle u, w, r\rangle \sim P_{\mathcal{G}}$. On the other hand, our generator is generalized, which means the fake node v can be latent and not found in $\mathcal{V}$.

To meet the two requirements, our generator should also employ relation-specific matrices (for relation-awareness), and generate samples from an underlying continuous distribution (for generalization). In particular, we leverage the following Gaussian distribution:

$$\mathcal{N}(\mathbf{e}_u^{G\top}\mathbf{M}_r^G, \sigma^2\mathbf{I}), \tag{7}$$

where $\mathbf{e}_u^G \in \mathbb{R}^{d\times 1}$ and $\mathbf{M}_r^G \in \mathbb{R}^{d\times d}$ denote the node embedding of $u \in \mathcal{V}$ and the relation matrix of $r \in \mathcal{R}$ for the generator. In other words, it is a Gaussian distribution with mean $\mathbf{e}_u^{G\top}\mathbf{M}_r^G$ and covariance $\sigma^2\mathbf{I} \in \mathbb{R}^{d\times d}$ for some choices of σ. Intuitively, the mean represents a fake node likely to be connected to u by relation r, and the covariance represents potential deviations. One naïve solution is to directly generate samples from $\mathcal{N}(\mathbf{e}_u^{G\top}\mathbf{M}_r^G, \sigma^2\mathbf{I})$. Nevertheless, as neural networks have shown strong ability in modeling complex structure [15], we integrate the multi-layer perceptron (MLP) into the generator for enhancing the expression of the fake samples. Hence, our generator is formulated as follows,

$$G(u, r; \theta^G) = f(\mathbf{W}_L \cdots f(\mathbf{W}_1\boldsymbol{e} + \mathbf{b}_1) + \mathbf{b}_L), \tag{8}$$

where we draw $\mathbf{e}$ from the distribution $\mathcal{N}(\mathbf{e}_u^{G\top}\mathbf{M}_r^G, \sigma^2\mathbf{I})$. Here $\mathbf{W}_*$ and $\mathbf{b}_*$ respectively denote the weight matrix and the bias vector for each layer, and f is an activation function. The parameter set of the generator is thus $\theta^G = \{\mathbf{e}_u^G : u \in \mathcal{V}, \mathbf{M}_r^G : r \in \mathcal{R}, \mathbf{W}_*, \mathbf{b}_*\}$, *i.e.,* the union of all node embeddings and relation matrices, as well as the parameters of MLP.

As motivated earlier, the generator wishes to fool the discriminator by generating close-to-real fake samples, such that the discriminator gives high score to them.

$$\mathcal{L}^G = \mathbb{E}_{\langle u,r\rangle\sim P_{\mathcal{G}}, \mathbf{e}'_v\sim G(u,r;\theta^G)} - \log D(\mathbf{e}'_v|u,r) + \lambda^G\|\theta^G\|_2^2, \tag{9}$$

where $\lambda^G > 0$ controls the regularization term. The parameters θ^G of the generator can be optimized by minimizing $\mathcal{L}^G$.

3.3 Optimization and Complexity Analysis

We adopt the iterative optimization strategy to train HeGAN. In each iteration, we alternate the training between the generator and discriminator. Specifically, we first fix θ^G, and generate fake samples to optimize θ^D and thus improve the discriminator. Next, we fix θ^D, and optimize θ^G in order to produce increasingly better fake samples as evaluated by the discriminator. We repeat the above

Algorithm 1 Model training for HeGAN

Require: HIN $\mathcal{G}$, number of generator and discriminator trainings per epoch n_G, n_D, number of samples n_s

Initialize θ_G and θ_D for G and D, respectively
while not converge **do**
  for $n = 0; n < n_D$ **do** ▷ **Discriminator training**
    Sample a batch of triples, *i.e.,* $\langle u, v, r\rangle \sim P_{\mathcal{G}}$
    Generate n_s fake nodes $\mathbf{e}'_v \sim G(u, r; \theta^G)$ for each $\langle u, r\rangle$
    Sample n_s relations $r' \sim P_{\mathcal{R}'}$ for each $\langle u, v\rangle$
    Update θ^D according to Eq. 6
  end for
  for $n = 0; n < n_G$ **do** ▷ **Generator training**
    Sample a batch of triples, *i.e.,* $\langle u, v, r\rangle \sim P_{\mathcal{G}}$
    Generate n_s fake nodes $\mathbf{e}'_v \sim G(u, r; \theta^G)$ for each $\langle u, r\rangle$
    Update θ^G according to Eq. 9
  end for
end while
return θ^G and θ^D

Table 2: Description of datasets.

Datasets	#Nodes	#Edges	#Node types	#Labels
DBLP	37,791	170,794	4	4
Yelp	3,913	38,680	5	3
Aminer	312,776	599,951	4	6
Movielens	10,038	1,014,164	5	N.A.

process for more iterations until our model converges. The model training for HeGAN is outlined in Algorithm 1.

We now conduct a complexity analysis. The updating of the generator and discriminator in each iteration mainly involve the updating of node vectors and relation matrices, whose time complexity is $O(n_s \cdot |\mathcal{V}| \cdot d^2)$, where n_s is the number of samples, $|\mathcal{V}|$ is the number of nodes and d is the embedding dimensionality. Hence, the complexity of training our model per iteration is $O((n_G + n_D) \cdot n_s \cdot |\mathcal{V}| \cdot d^2)$, where n_G and n_D are the number of generator and discriminator training per iteration, respectively. Since we treat n_G, n_D, n_s and d as small constants, the complexity of each iteration in our model is linear with the number of nodes in the HIN, *i.e.,* $|\mathcal{V}|$, indicating the efficiency and scalability of HeGAN.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of HeGAN on a wide range of tasks, including node clustering and classification, as well as link prediction and recommendation. We further analyze the model performance in the contexts of the underlying mechanism, efficiency and parameter sensitivity.

4.1 Experimental Setup

4.1.1 Datasets. We conduct extensive experiments on four benchmark datasets [12, 15], namely DBLP (with four types of nodes: *author*, *paper*, *conference* and *term*), Yelp (*user*, *business*, *service*, *star* and *reservation*), AMiner (*author*, *paper*, *conference* and *reference*), and Movielens (*user*, *movie*, *age*, *occupation* and *type*). We organize them into HINs, as summarized in Table 2. On the first three datasets, we perform node classification and clustering *w.r.t.* the given labels, as well as link prediction. Movielens is reserved for a recommendation task. Task descriptions will be postponed until their respective experiments.

Table 3: Performance comparison on node clustering. (bold: best; underline: runner-up)

Methods	DBLP	Yelp	AMiner
Deepwalk	0.7398	0.3306	<u>0.4773</u>
LINE-1st	0.7412	0.3556	0.3518
LINE-2nd	0.7336	0.3560	0.2144
GraphGAN	0.7409	0.3413	-
ANE	0.7138	0.3145	0.4483
HERec-HNE	0.7274	0.3476	0.4635
HIN2vec	0.7204	0.3185	0.2812
Metapath2vec	<u>0.7675</u>	<u>0.3672</u>	0.4726
HeGAN	**0.7920****	**0.4037****	**0.5052****

4.1.2 Baselines. We consider three categories of network embedding methods: traditional (Deepwalk, LINE), GAN-based (GraphGAN, ANE) and HIN (HERec-HNE, HIN2vec, Metapath2vec) embedding algorithms.

- **Deepwalk** [23] performs truncated random walks on networks, and employs the skip-gram model.
- **LINE** [31] exploits the first (**LINE-1st**) and second (**LINE-2nd**) order proximity in networks.
- **GraphGAN** [33] unifies generative and discriminative learning with GAN in a minimax game.
- **ANE** [9] adopts the adversarial learning principle to push the latent representations towards a fixed posterior distribution.
- **HERec-HNE** [27] designs a type-constrained strategy to filter node sequences for semantic-preserving HIN embedding.
- **HIN2vec** [12] learns latent representations of both nodes and meta-paths jointly, and thus preserves the semantics of HINs.
- **Metapath2vec** [11] samples meta-path-based random walks for semantic-preserving HIN embedding.

4.1.3 Implementation. Working details, including hyperparameter settings, will be discussed in the reproducibility supplement. We will also study the impact of hyperparameters in HeGAN in Sect. 4.3.4.

4.1.4 Significance Test. For tabular results in Tables 3, 4 and 5, we use ** (or *) to indicate that HeGAN is significantly different from the runner-up method based on paired t-tests at the significance level of 0.01 (or 0.05).

4.2 Experimental Analysis

In this section, we empirically compare HeGAN and the baselines on various downstream tasks. Note that we do not report the results of GraphGAN on the AMiner dataset, since it cannot run on this large dataset even on a machine with 256GB memory.

4.2.1 Node Clustering. We employ the K-Means algorithm to perform clustering, and evaluate the clustering quality in terms of *normalized mutual information* (NMI) *w.r.t.* the node labels.

Table 4: Performance comparison on node classification. (bold: best; underline: runner-up)

Methods	DBLP			Yelp			AMiner		
	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1	Accuracy
Deepwalk	0.9201	0.9242	0.9298	0.8262	0.7551	0.8145	0.9519	0.9460	0.9529
LINE-1st	0.9239	0.9213	0.9285	0.8229	0.7440	0.8126	0.9776	0.9713	0.9788
LINE-2nd	0.9144	0.9172	0.9236	0.7591	0.5518	0.7571	0.9469	0.9341	0.9471
GraphGAN	0.9198	0.9210	0.9286	0.8098	0.7268	0.7820	-	-	-
ANE	0.9143	0.9153	0.9189	0.8232	0.7623	0.7932	0.9256	0.9203	0.9221
HERec-HNE	0.9214	0.9228	0.9299	0.7962	0.7713	0.7912	0.9801	0.9726	0.9784
HIN2vec	0.9141	0.9115	0.9224	0.8352	0.7610	0.8200	0.9799	0.9775	0.9801
Metapath2vec	0.9288	0.9296	0.9360	0.7953	0.7884	0.7839	0.9853	0.9860	0.9857
HeGAN	**0.9381****	**0.9375****	**0.9421****	**0.8524****	**0.8031****	**0.8432****	**0.9864***	**0.9873***	**0.9883***

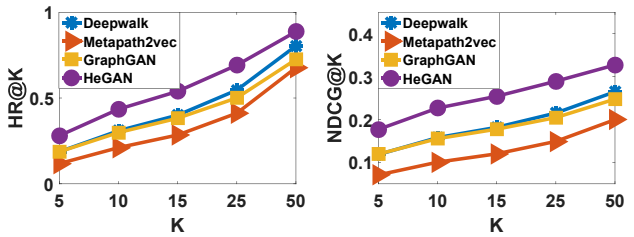
Based on the results in Table 3, we make the following observations. (i) HeGAN is consistently better than all the baselines, demonstrating its effectiveness for learning representations in HINs. (ii) Among the baselines, HIN embedding methods (*i.e.,* HERec-HNE, HIN2vec and Metapth2vec) generally outperform traditional methods, implying the importance of semantic-preserving embedding. (iii) HeGAN significantly outperforms both HIN and GAN-based (*i.e.,* ANE and GraphGAN) embedding methods, showing that our model can learn semantic-preserving representations in a robust manner through the adversarial principle. Overall, HeGAN achieves performance gains over the best baseline by 3%, 10% and 6% on the three datasets, respectively.

4.2.2 Node Classification. We use 80% of the labeled nodes to train a logistic regression classifier, and test the classifier on the remaining 20% nodes.

Since the tasks are multi-class, We report *Macro-F1*, *Micro-F1* and *Accuracy* on the test set in Table 4. Similar conclusions to the node clustering task can be drawn, where HeGAN consistently outperforms the best baseline with statistical significance. It is also worth noting that our performance margins over the best baseline becomes smaller compared to the node clustering task, since in classification all methods are helped by the supervision, narrowing their gaps.

4.2.3 Link Prediction. In this task, we predict *user-business* links on Yelp, and *author-paper* links on DBLP and AMiner. We randomly hide 20% of such links from the original network as the ground truth positives, and randomly sample disconnected node pairs of the given form as negative instances. The ground truth serves as our test set. Subsequently, we utilize the residual network to learn node representations. After obtaining the node representations, we adopt two ways to perform link prediction, namely, *logistic regression* and *inner product*. To train the logistic regression classifier, we used links and negative instances from the residual network without overlapping with the test set.

Since this is a binary task, we adopt *Accuarcy*, *AUC* and *F1* as evaluation metrics, as shown in Table 5. Again, similar conclusions can be drawn, where HeGAN consistently and significantly outperform all the baselines, except for one case. We further observe that, the performance lead of HeGAN over the best baseline is much larger with inner product than with logistic regression. In particular, in terms of F1 scores, with inner product we outperform the best baseline by 6%, 7% and 11% on the three datasets, respectively, whereas with logistic regression the improvement is only about 2%. We hypothesize that HeGAN innately learns a much better structure- and semantics-preserving embedding space than the baseline methods, since the inner product only relies on the learnt representations without resorting to any external supervision.



(a) HR (b) NDCG

Figure 2: Performance comparison on the Movielens recommendation task.

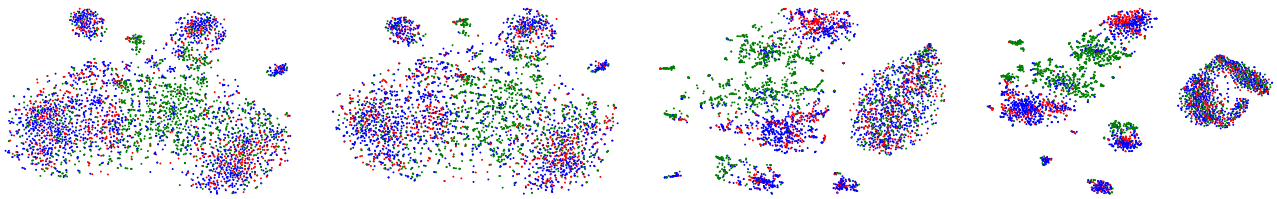
4.2.4 Recommendation. We adopt the Movielens dataset for the recommendation task. For each user, we aim to recommend a set of unseen movies which may be liked by the user. Following [15], we adopt the leave-one-out method for evaluation, *i.e.,* the last movie of each user is held out for testing, and the remaining data are used for training the node embeddings. During testing, we select top-K movies with the highest inner product with the target user, and evaluate the recommendation performance with *HR@K* (hit ratio) and *NDCG@K*.

We report the results of HeGAN and three representative baselines, namely, Deepwalk, GraphGAN and Metapath2vec. From Fig. 2, we observe that HeGAN consistently outperforms all the baselines across all K values on both evaluation metrics. For example, HeGAN outperforms the baselines by 41~106% in HR@10, and 44~126% in NDCG@10. This task shows the effectiveness of HeGAN on a ranking-based objective. It is worth noting that Metapth2vec is not performing well in this task. One hypothesis is that there is a lack of meaningful meta-paths in the Movielens dataset.

4.2.5 Network Visualization. To examine the network representations intuitively, we visualize the embeddings of *business* nodes in Yelp using the t-SNE [20] algorithm in Fig. 3. From the plots,

Table 5: Performance comparison on link prediction. (bold: best; underline: runner-up)

	Methods	DBLP			Yelp			AMiner		
		Accuracy	AUC	F1	Accuracy	AUC	F1	Accuracy	AUC	F1
Logistic Regression	Deepwalk	0.5441	0.5630	0.5208	0.7161	0.7825	0.7182	0.4856	0.5182	0.4618
	LINE-1st	0.6546	0.7121	0.6685	0.7226	0.7971	0.7099	0.5983	0.6413	0.6080
	LINE-2nd	0.6711	0.6500	0.6208	0.6335	0.6745	0.6499	0.5604	0.5114	0.4925
	GraphGAN	0.5241	0.5330	0.5108	0.7123	0.7625	0.7132	-	-	-
	ANE	0.5123	0.5430	0.5280	0.6983	0.7325	0.6838	0.5023	0.5280	0.4938
	HERec-HNE	0.7123	0.7823	0.6934	0.7087	0.7623	0.6923	0.7089	0.7776	0.7156
	HIN2vec	0.7180	0.7948	0.7006	0.7219	0.7959	0.7240	0.7142	0.7874	0.7264
	Metapath2vec	0.5969	0.5920	0.5698	0.7124	0.7798	0.7106	0.7069	0.7623	0.7156
	HeGAN	**0.7290****	**0.8034****	**0.7119****	**0.7240****	**0.8075****	**0.7325****	**0.7198****	**0.7957****	**0.7389****
Inner Product	Deepwalk	0.5474	0.7231	0.6874	0.5654	0.8164	0.6953	0.5309	0.6064	0.6799
	LINE-1st	0.6647	0.7753	0.7363	0.6769	0.7832	0.7199	0.6113	0.6899	0.7123
	LINE-2nd	0.4728	0.4797	0.6325	0.4193	0.7347	0.5909	0.5000	0.4785	0.6666
	GraphGAN	0.5532	0.6825	0.6214	0.5702	0.7725	0.6894	-	-	-
	ANE	0.5218	0.6543	0.6023	0.5432	0.7425	0.6324	0.5421	0.6123	0.6623
	HERec-HNE	0.5123	0.7473	0.6878	0.5323	0.6756	0.7066	0.6063	0.6912	0.6798
	HIN2vec	0.5775	0.8295	0.6714	0.6273	**0.8340**	0.4194	0.5348	0.6934	0.6824
	Metapath2vec	0.4775	0.6926	0.6287	0.5124	0.6324	0.6702	0.6243	0.7123	0.6953
	HeGAN	**0.7649****	**0.8712****	**0.7837****	**0.7391****	0.8298	**0.7705****	**0.6505****	**0.7431****	**0.7752****



(a) Deepwalk (b) GraphGAN (c) Metapath2vec (d) HeGAN

Figure 3: Visualization of representative baselines and HeGAN on Yelp. Color indicates the category of business.

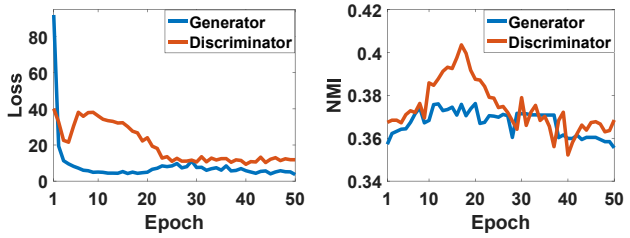
we find that Deepwalk and GraphGAN cannot effectively identify different business categories due to the ignorance of heterogeneity. On the other hand, Metapath2vec and HeGAN both can reasonably separate the business categories, while HeGAN has a more crisp boundary and denser clusters.

4.3 Model Analysis on HeGAN

Next, we perform a series of analysis to better understand the underlying mechanism of HeGAN, as well as its efficiency and hyperparameter choices.

4.3.1 Adversarial Learning. First of all, in Fig. 4 we present the learning curves of the generator and the discriminator of HeGAN on Yelp, in terms of the change in the loss and the clustering quality. After some initial fluctuations in their losses, the generator and the discriminator starts to play a minimax game, gradually decreasing the losses of both. After about 20 epochs of adversarial training, the losses of both tend to converge, and the winner (*i.e.,* the discriminator) achieves a better performance. Note that when more epochs are trained, the clustering quality decreases due to overfitting.

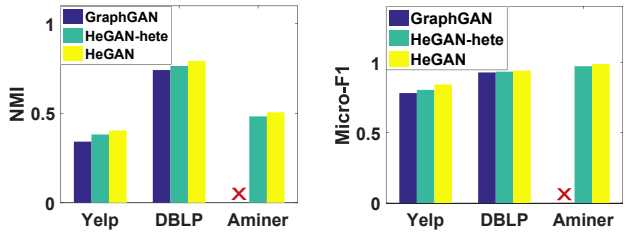
4.3.2 Heterogeneity and Generalized Generator. In this section, we study if HeGAN can effectively utilize heterogeneous information



(a) Loss change (b) Performance change in node clustering

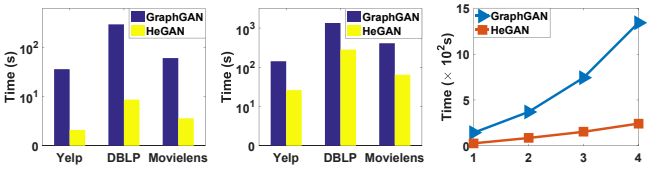
Figure 4: Learning curves of HeGAN on Yelp.

in HINs, and also if our generalized generator is effective. In particular, we compare HeGAN with GraphGAN (the closest baseline to ours) and a variant of our model named HeGAN-hete (*i.e.,* HeGAN without heterogeneous information, by assuming that all relations belong to the same type). We report their performance on node clustering and classification in Fig. 5. We find that the overall performance order is HeGAN > HeGAN-hete > GraphGAN. Such an ordering is not surprising, and reveals two major implications. First,



(a) Node clustering (b) Node classification

Figure 5: Impact of heterogeneity and generalized generator.



(a) Generator training (b) Overall training (c) Scalability

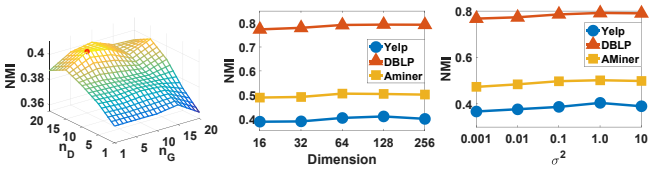
Figure 6: Efficiency evaluation on three datasets. (a) Time per generator training in log scale. (b) Time per training epoch in log scale. (c) Scalability study *w.r.t.* the number of nodes ($\times 10^4$) in linear scale.

different types of nodes and relations in HINs should be distinguished, as disregarding such information leads to the worse performance of HeGAN-hete compared to HeGAN . Second, between HeGAN-hete and GraphGAN, as both do not utilize heterogeneous information, their key difference lies in their generators. In particular, our model is generalized, in the sense that it generates fake, potentially latent, samples from a continuous Gaussian distribution, whereas GraphGAN generates only pick fake nodes from the original network. The result shows that our generalized generator can indeed produce more representative samples.

4.3.3 Efficiency. As mentioned above, our generalized generator draw samples directly from a continuous distribution, which significantly accelerate the training of generators. In contrast, GraphGAN employs Graph Softmax in its generator—although it is much more efficient than the normal softmax, it is not as fast as the generator of HeGAN. We report the updating time per generator training and per overall training epoch in Fig. 6(a) and (b), respectively. As shown, HeGAN runs about ten times faster than GraphGAN for both generator and overall training, demonstrating that our generator is not only effective, but also efficient. (Note that GraphGAN cannot run on AMiner due to memory constraint.)

We also conduct a scalability study, by sampling a few smaller sub-networks from Aminer. The results are shown in Fig. 6(c), where the running time of HeGAN increases linearly *w.r.t.* the number of nodes, consistent with our complexity analysis in Sect. 3.3.

4.3.4 Parameter Analysis. Finally, we investigate the impact of the parameters in the following categories, using the node clustering quality as a reference. Similar patterns have been observed on the performance of other tasks.



(a) Epoch (b) Dimension (c) Variance

Figure 7: Impact of parameters on HeGAN.

First, the number of generator and discriminator training per epoch, n_G and n_D. They control the balance between the two players. We run a grid search over $\{1, 5, 10, 15, 20\}^2$ and plot their performance in Fig. 7(a). HeGAN achieves the optimal performance near $(5, 15)$, *i.e.,* $n_G = 5, n_D = 15$. In general, the performance is stable when the discriminator is trained more than the generator, *i.e.,* when $n_D \geq 10$ and $n_G \leq 10$. This observation is consistent with previous findings [1].

Second, the embedding dimension, d. As shown in Fig. 7(b), overall our model is not sensitive to this parameter. Nevertheless, the optimal performance is obtained when d is 64 or 128.

Third, the variance of Gaussian distribution, σ^2. As shown in Fig. 7(c), our model achieves optimal performance when $\sigma^2 = 1.0$ and is generally stable around that value, although too small or large values (such as $\sigma^2 < 0.1$) would harm the model.

5 RELATED WORK

We review the most related work in network embedding, HIN embedding and generative adversarial networks.

Network embedding. Network embedding [3, 8] has shown its potential to learn structure-preserving node representations, and has been successfully applied to many data mining tasks. Contemporary methods usually explore network topology as context information, based on random walks (*e.g.,* Deepwalk [23] and node2vec [14]), neighborhoods (*e.g.,* LINE [31] and SDNE [32]) or high-order proximity (*i.e.,* GraRep [5], NEU [36] and AROPE [40]). Unfortunately, these methods only deal with homogeneous networks, and thus they cannot learn semantic-preserving representations in HINs. Meanwhile, recently emerged graph neural networks (*e.g.,* GCN [18]) are proposed for representation learning in an end-to-end manner with task-specific supervision. Their goals differ from our scope, which aims to learn node presentations in an unsupervised manner to support arbitrary downstream tasks.

HIN embedding. Recently, heterogeneous information networks (HIN) [28] have been proposed to model complex entities and their rich relations in various applications [7, 16, 17, 27, 34, 39]. To marry the advantages of HIN and network embedding, numerous methods [11, 26, 29, 30] have been proposed for representation learning in HINs. One major line of work leverages meta-path-based contexts for semantic-preserving embedding, including meta-path-based similarity [26] and meta-path-based neighbors [11]. Note that these methods rely on domain knowledge to choose the right meta-paths, whereas there also exist a few methods [6, 12, 29, 30] which do not require meta-path selection. Moreover, task-specific embedding learning in HINs [7, 27, 34] has also been explored recently, which

departs from our goal to learn structure- and semantic-preserving representations to support arbitrary tasks.

Generative adversarial networks. Generative adversarial networks (GANs) [13] have demonstrated superior performance in many problems [19, 35, 37]. They hinge on the principle of adversarial learning, where a generator and discriminator compete with each other to improve their outcomes. Inspired by GANs, a few studies [4, 9, 22, 33, 38] leverage the adversarial principal to learn more robust representations. Some of them [4, 9, 22, 38] impose a fixed prior distribution on the embedding space to enhance the robustness of the learned representations. However, these methods ignore the heterogeneity of nodes and relations, and thus cannot capture the rich semantics on HINs.

6 CONCLUSION

In this paper, we proposed a novel framework called HeGAN for HIN embedding based on the adversarial principle. We elaborately designed the relation-aware discriminator and generator to fit the heterogeneous setting. Specifically, *w.r.t.* to a given relation, the discriminator can tell whether a node pair is real or fake, whereas the generator can produce fake node pairs that mimic real pairs. To further improve the effectiveness and efficiency of sample generation, we proposed a generalized generator, which is able to directly sample latent nodes from a continuous distribution. Expensive experimental results have verified the effectiveness and efficiency of HeGAN on various tasks.

7 ACKNOWLEDGEMENTS

This research was supported by the National Natural Science Foundation of China (No. 61772082, 61702296), the National Key Research and Development Program of China (2017YFB0803304), the Beijing Municipal Natural Science Foundation (4182043) and the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Approval No. 18-C220-SMU-006).

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*. 214–223.
[2] Sezin Kircali Ata, Yuan Fang, Min Wu, Xiao-Li Li, and Xiaokui Xiao. 2017. Disease gene classification with metagraph representations. *Methods* 131 (2017), 83–92.
[3] Hongyun Cai, Vincent W Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[4] Xiaoyan Cai, Junwei Han, and Libin Yang. 2018. Generative Adversarial Network Based Heterogeneous Bibliographic Network Representation for Personalized Citation Recommendation.. In *AAAI*. 5747–5754.
[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. 891–900.
[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *SIGKDD*. 119–128.
[7] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*. 1177–1186.
[8] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[9] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *AAAI*. 2167–2174.
[10] Ming Ding, Jie Tang, and Jie Zhang. 2018. Semi-supervised learning on graphs with generative adversarial nets. In *CIKM*. 913–922.
[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. 135–144.
[12] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. 1797–1806.
[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.
[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[16] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*. 1531–1540.
[17] Binbin Hu, Zhiqiang Zhang, Chuan Shi, Jun Zhou, Xiaolong Li, and Yuan Qi. 2019. Cash-out User Detection based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism. In *AAAI*.
[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[19] Yan Li and Jieping Ye. 2018. Learning Adversarial Networks for Semi-Supervised Text Classification via Policy Gradient. In *SIGKDD*. 1715–1723.
[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008), 2579–2605.
[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop*.
[22] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding.. In *IJCAI*. 2609–2615.
[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.
[24] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*.
[25] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *ICML*. 1060–1069.
[26] Jingbo Shang, Meng Qu, Jialu Liu, Lance M Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769* (2016).
[27] Chuan Shi, Binbin Hu, Xin Zhao, and Philip Yu. 2018. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.
[28] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 17–37.
[29] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. 2018. AspEm: Embedding Learning by Aspects in Heterogeneous Information Networks. In *SDM*. 144–152.
[30] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *SIGKDD*. 1165–1174.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[32] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *SIGKDD*. 1225–1234.
[33] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGan: Graph representation learning with generative adversarial nets. In *AAAI*. 2508–2515.
[34] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: signed heterogeneous information network embedding for sentiment link prediction. In *WSDM*. 592–600.
[35] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*. 515–524.
[36] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast network embedding enhancement via high order proximity approximation. In *IJCAI*. 19–25.
[37] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.
[38] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. 2018. Learning deep network representations with adversarially regularized autoencoders. In *SIGKDD*. 2663–2671.
[39] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi, and Yangyong Zhu. 2018. Deep collective classification in heterogeneous information networks. In *WWW*. 399–408.
[40] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *SIGKDD*. 2778–2786.
[41] Tom Chao Zhou, Hao Ma, Michael R Lyu, and Irwin King. 2010. UserRec: A User Recommendation Framework in Social Tagging Systems. In *AAAI*. 1486–1491.

A IMPLEMENTATION DETAILS

A.1 Environment

We implement the proposed model using the Python library Tensorflow[1]. All the experiments are conducted on a Linux server with one GPU (GeForce RTX) and CPU (Intel Xeon W-2133). We release the source code of HeGAN at https://github.com/librahu/HeGAN. And the data in the paper is available at https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding

A.2 Parameter Settings for HeGAN

We perform Adaptive Moment Estimation (Adam) to optimize our model with learning rate 0.0001. Moreover, we set the regularization parameters for the generator and discriminator to $\lambda_D = \lambda_G = 0.0001$, and utilize a two-layer MLP with a ReLU activation function for the generator. For each iteration of generator and discriminator training, we use a batch size of 32 and set the number of samples to $n_s = 16$. We run $n_G = 5$ iterations of generator training and $n_D = 15$ iterations of discriminator training in each epoch. Careful initialization of neural network model is crucial to avoid poor local minima. Hence, HeGAN initializes initial embeddings in generator and discriminator from a pretrained embedding model (Metapath2vec).

A.3 Parameter Settings for Baselines

The embedding dimensions for all methods are set as 64. For random walk-based methods (*i.e.,* Deepwalk, HIN2vec and Metapath2vec), we set the number of walks per node to $w = 10$, the walk length to $l = 100$ and the window size to $\tau = 5$. For HIN embedding methods (*i.e.,* HERec-HNE and Metapath2vec), we select the meta-paths APCPA, UBStBU, APCPA and UMTMU in DBLP, Yelp, AMiner and Movielens, respectively, since they have the best performance under these meta-paths. For LINE, we set the number of samples as 10000. For GraphGAN, we set the learning rate as 0.001 and the number of samples as 20. For ANE, we follow the parameter settings in the original paper.

[1] https://www.tensorflow.org/